

Práctica 1: Boletín de ejercicios

Procesamiento de Imagen, Vídeo y Audio

Grado en Ciencia e Ingeniería de Datos

Curso 2022/2023

La práctica consiste en la realización de una serie de ejercicios de procesamiento de imágenes, haciendo uso de técnicas vistas en la asignatura. Los ejercicios del curso se codificarán en **Python 3.x** con las bibliotecas de procesamiento **Numpy**, **SciPy** y **Scikit-image**.

- Las práctica deberá realizarse de forma **individual**
- Se deberá **responder a los ejercicios en un Notebook**, mezclando bloques de código y visualización con respuestas a las preguntas en formato Markdown. Se pueden usar ficheros `.py` externos si se considera conveniente.
- No está permitido el uso de versiones de Python 2.x.

Entrega y evaluación

- El alumnado deberá entregar en el Campus Virtual (Moodle de la UDC), en un repositorio habilitado para tal fin:
 - El código de las prácticas y las respuestas a los ejercicios en formato **Notebook** (fichero `.ipynb`), junto con el resto de ficheros `.py` de la práctica.
 - Un fichero de texto (`python.version.txt`) indicando la versión de Python utilizada
 - Un fichero (`requirements.txt`), en formato compatible con **pip**, indicando las librerías utilizadas y sus versiones.
 - El contenido del notebook, con todos los resultados ejetuados y mostrados, guardado en formato PDF (preferiblemente) o HTML.
- El material entregado debe estar subido a la plataforma Moodle. No se aceptarán entregas a través de enlaces a otros servicios de almacenamiento en la nube.
- La fecha límite de entrega, entorno a Semana Santa, será especificada y notificada en el Campus Virtual.
- El boletín consta de 20 ejercicios, cada uno de los cuales supone una nota máxima de 0,5 puntos sobre un total de 10 puntos. El profesor de prácticas podrá citar al alumnado para efectuar una defensa oral de la práctica si lo considera conveniente.
- El alumnado que presente códigos y/o respuestas con indicios de plagio (en cualquiera de las respuestas y con respecto a otras prácticas presentadas este curso o en cursos anteriores, o recursos online) obtendrá una calificación de Suspenso en la parte práctica (calificado con 0.0), independientemente de la nota que pudiera merecer la calidad de la práctica, e independientemente de si las personas implicadas resultan ser plagiadas o plagiadoras.

Consideraciones previas

1. Se recomienda instalar las **últimas versiones estables** de las bibliotecas aplicables a la versión de Python seleccionada, utilizando un gestor de paquetes (pip o conda).
2. Se recomienda usar un entorno de **Virtual Environment** (e.g. `venv`) que permita gestionar adecuadamente las dependencias de paquetes y reproducir el entorno de ejecución. No usar un Virtual Environment no exime de llevar un control de las versiones de Python y librerías usadas, así como facilitarlas en la entrega.
3. Se recomienda seguir las **guías de estilo estándar PEP 8** ([PEP 8 Style Guide for Python Code](https://www.python.org/dev/peps/pep-0008/)¹) de codificación en Python que sean aplicables.

Librerías mínimas recomendadas

Para la realización de la práctica se necesitará, como mínimo, instalar y consultar parte de la documentación de las siguientes librerías:

- [Scikit-image](https://scikit-image.org/)². Librería de procesamiento de imágenes, basada en **NumPy** y **SciPy**, en la que centraremos las prácticas. Nombre del paquete: `scikit-image`.
- [NumPy](https://numpy.org/doc/)³. Librería de operaciones matriciales. Se instalará automáticamente como dependencia de **Scikit-image**.
- [SciPy](https://scipy.org/)⁴. Librería extensible de algoritmos de computación científica basados en **NumPy**. Es de especial relevancia para esta práctica el módulo `scipy.ndimage`⁵. **SciPy** Se instalará automáticamente como dependencia de **Scikit-image**.
- [Matplotlib](https://matplotlib.org/stable/index.html)⁶. Librería de visualización científica compatible con **NumPy**. Será necesaria para visualizar imágenes y gráficas. También la usaremos para lectura y escritura de imágenes. Nombre del paquete: `matplotlib`.

Virtual Environments y Jupyter Notebooks

El uso de Virtual Environments es habitual en el desarrollo de software usando Python, pues permite gestionar de forma sencilla los siguientes retos:

- Mantener múltiples instalaciones de Python en el sistema con diferentes versiones del lenguaje y diferentes versiones de librerías.
- Poder controlar la versión concreta de Python y de las librerías utilizadas, independientemente de las disponibles a nivel global en el sistema.
- Reproducir el entorno de ejecución exacto en cualquier otro sistema.

Esto es útil para nuestro caso concreto, pues permitirá mantener estable el entorno de desarrollo durante todo el curso (independientemente de posibles actualizaciones del sistema), así como establecer mecanismos adecuados para que los profesores puedan ejecutar las prácticas en condiciones reproducibles con respecto a las instalaciones del alumnado.

Para reproducir un entorno Python, típicamente necesitaremos conocer dos cosas:

¹PEP 8 Style Guide for Python Code: <https://www.python.org/dev/peps/pep-0008/>

²Scikit-image: <http://scikit-image.org/>

³NumPy: <https://numpy.org/doc/>

⁴SciPy: <https://scipy.org/>

⁵scipy.ndimage: <https://docs.scipy.org/doc/scipy/reference/ndimage.html>

⁶Matplotlib: <https://matplotlib.org/stable/index.html>

Versión de Python. Para la usar una versión concreta de Python necesitaremos disponer de una instalación de Python con esa versión en el sistema. Por lo general, dependiendo del sistema, podremos tener disponibles diferentes versiones con varios nombres (e.g. `python`, `python3`, `python3.6`, `python3.7`, `python3.8`, etc.). Podremos consultar la versión de cualquier intérprete python usando la opción `--version` por línea de comandos. Por ejemplo, para almacenar la versión en un fichero:

```
$ python --version > python_version.txt
```

Lista de librerías y sus versiones. Típicamente gestionaremos las librerías en una instalación de Python usando un gestor de paquetes tipo `pip`⁷. Estos gestores permiten instalar paquetes de uno en uno (`pip install ...`), especificando la lista de nombres. Esto sería lo habitual, a medida que vamos desarrollando y requiriendo funcionalidades. En cualquier caso, siempre podremos obtener una foto fija de la instalación actual usando `pip freeze` y guardarla en disco en un fichero, normalmente llamado `requirements.txt`:

```
$ pip freeze > requirements.txt
```

Y para instalar todos los paquetes especificados en un fichero previamente almacenado (se entiende que en otro sistema), podremos hacerlo mediante:

```
$ pip install -r requirements.txt
```

Configuración y uso de Virtual Environments con `venv`

Python dispone de un entorno estándar de Virtual Environments (`venv`⁸) desde su versión 3.3, que explicaremos a continuación. Si bien, existen opciones equivalentes con librerías alternativas, o en distribuciones Python propietarias como Anaconda. El alumnado puede seguir estas indicaciones o consultar la documentación de los sistemas alternativos no estándar que desee utilizar, aunque muchas de las indicaciones son equivalentes y con pocos cambios.

Crear `venv`. Asumiendo que la versión concreta de python que queremos usar se encuentra disponible en el sistema bajo el nombre `python3.7`, podemos crear un nuevo Virtual Environment vacío, en el directorio actual (e.g. `/GCED/PIVA/P1/`), llamado `piva`.

```
$ cd ~/GCED/PIVA/P1
$ python3.7 -m venv piva
```

Activar y desactivar `venv`. Para entrar, o salir, del Virtual Environment, podemos usar los scripts `activate` y `deactivate`, respectivamente, disponibles en la carpeta local (llamada `piva`, en este caso). Dentro del Virtual Environment, tanto el comando `python`,

⁷`pip`: <https://pip.pypa.io/en/stable/>

⁸`venv`: <https://docs.python.org/3/library/venv.html>

como otros asociados a la instalación (e.g. `pip`), y las librerías disponibles, serán aquellas propias del Virtual Environment, y no las del sistema. Por ejemplo, podemos probar `pip freeze` para verificar que no hay paquetes instalados.

```
# Entramos en 'piva'
$ source piva/bin/activate
(piva) $ _

# Ahora estamos dentro de 'piva'. Veamos, por ejemplo, dónde
→ está 'python', cual es su versión, dónde está 'pip' y qué paquetes
→ hay instalados:
(piva) $ which python
(piva) $ python --version
(piva) $ which pip
(piva) $ pip freeze

# Salimos de 'piva'
(piva) $ deactivate
$ _

# Ahora podríamos volver a probar lo anterior, para verificar que ya
→ estamos, otra vez, en la instalación del sistema
```

Instalación de paquetes en `venv`. Dentro del Virtual Environment, tras haber usado el script `activate`, los comandos de gestión de paquetes tendrán efecto sobre la instalación del `venv`. Es **altamente recomendable**, antes de nada, sobre una instalación limpia, actualizar `pip`, así como instalar `setuptools` y `wheel`. Este último permite acelerar la instalación de paquetes evitando la compilación. A continuación, podemos instalar la lista de paquetes, a mano, por ejemplo `scikit-image` y `matplotlib` (alternativamente mediante un fichero de `requirements.txt`).

```
(piva) $ pip install -U pip wheel setuptools
# Instalamos las librerías (instalará sus dependencias):
(piva) $ pip install scikit-image matplotlib
# Alternativamente, si disponemos de un fichero requirements.txt para
→ reproducir un entorno previo:
(piva) $ pip install -r requirements.txt
```

Uso de Virtual Environments en Jupyter Notebooks

En general, si tenemos un script de Python que queremos ejecutar en un Virtual Environment, bastará con activar (con `activate`) el entorno, e invocar el comando `python` local.

No obstante, en Jupyter Notebook, el sistema de ejecución es diferente, ya que el entorno Notebook hace uso de "kernels" (IPython Kernel, para ser precisos), sobre los que se delega cada ejecución del programa. Por lo general, los entornos Python del sistema estarán disponibles por defecto en nuestra instalación de Notebook, pero esto no ocurrirá con nuestros Virtual Environments.

Para instalar un Virtual Environment como un Kernel de IPython o Jupyter, debemos activar el entorno, e invocar el comando indicado a continuación.

```
$ cd ~/GCED/PIVA/P1/  
$ source piva/bin/activate  
(piva) $ python -m ipykernel install --user --name=Piva  
(piva) $ deactivate  
$ _
```

Tras la instalación es posible salir del entorno, y tendremos un nuevo Kernel disponible en Notebook, en este caso con el nombre "Piva". Encontraremos el Kernel en el menú **Kernel** > **Change kernel** > ... de la aplicación.

El paso de instalación del kernel solamente será necesario hacerlo una vez, para que Jupyter tenga constancia de este Virtual Environment y lo pueda usar. Los cambios efectuados en el Virtual Environment, como pueden ser instalación de paquetes nuevos, tendrán efecto directo sobre este kernel. Aunque será necesario reiniciar el kernel para que surtan efecto.

Podemos ver y administrar la lista de kernels disponibles en Jupyter usando la utilidad `jupyter kernelspec` (seguido por e.g. `--help`, `list`, `uninstall <name>`, etc.). Por ejemplo:

```
$ jupyter kernelspec list
```

1. Entrada, salida y representación de imágenes

Entrada/salida recomendada

La librería **Scikit-image**, cuenta con un módulo de entrada, salida y visualización de datos de imagen [skimage.io](https://scikit-image.org/docs/dev/api/skimage.io.html)⁹.

Es relevante tener en cuenta que el módulo **skimage.io** utiliza diferentes implementaciones de cada una de sus funciones, basándose en librerías externas que sirven de *plugins*, cuya documentación debemos consultar para comprender las funcionalidades avanzadas. Podemos ver las opciones disponibles, así como la preferencia para cada función, mediante:

```
import skimage.io as io
# Lista de plugins disponibles (no necesariamente instalados):
io.find_available_plugins()
# Lista de plugins instalados para cada función:
io.plugin_order()
```

En esta práctica, recomendamos el uso de **skimage.io** para leer ([io.imread](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imread)¹⁰) y escribir ([io.imwrite](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imwrite)¹¹) ficheros de imagen, que por defecto estarán soportados por la librería [imageio](https://imageio.readthedocs.io/en/stable/reference/userapi.html)¹². Para la visualización de imágenes usaremos [io.imshow](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imshow)¹³, que por defecto estará implementada por [pyplot.imshow](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html)¹⁴. En ocasiones será conveniente usar directamente esta última, o usar sus parámetros específicos, para tener mayor control sobre la visualización.

Representación numérica de imágenes

En **Scikit-image**, así como en otras librerías de procesamiento de imagen en Python, las imágenes vienen representadas por matrices NumPy ([np.array](https://numpy.org/doc/stable/reference/generated/numpy.array.html)¹⁵). Existen convenciones sobre la forma e índices de las matrices ([np.array.shape](https://numpy.org/doc/stable/reference/generated/numpy.array.shape), típicamente [filas, columnas, canales] para imágenes 2D), así como sobre el tipo de datos ([np.array.dtype](https://numpy.org/doc/stable/reference/generated/numpy.array.dtype)) y su rango de valores (típicamente con valores restringidos al rango [0,1] para tipos flotantes, y al rango completo de enteros positivos para tipos enteros, e.g. [0,255] para [np.uint8](https://numpy.org/doc/stable/reference/generated/numpy.uint8.html)). Esto tiene implicaciones relevantes en la forma en la que las funciones de la librería interpretan los datos. Revise la [Guía de Usuario](https://scikit-image.org/docs/dev/user_guide.html)¹⁶ para obtener información detallada.

Se recomienda el uso de representaciones flotantes en el rango [0, 1] para hacer operaciones con imágenes, salvo causas debidamente justificadas.

Imágenes de prueba

A lo largo de la práctica, usaremos imágenes de prueba de tres fuentes distintas:

- Leyendo ficheros de imagen, típicamente en formato **.png**, no comprimido.
- Construyendo imágenes artificiales mediante manipulación directa de matrices.
- Invocando las funciones del repositorio de pruebas de **Sciki-image** ([skimage.data](https://scikit-image.org/docs/stable/api/skimage.data.html)¹⁷).

⁹ [skimage.io: https://scikit-image.org/docs/dev/api/skimage.io.html](https://scikit-image.org/docs/dev/api/skimage.io.html)

¹⁰ [io.imread: https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imread](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imread)

¹¹ [io.imwrite: https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imwrite](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imwrite)

¹² [imageio: https://imageio.readthedocs.io/en/stable/reference/userapi.html](https://imageio.readthedocs.io/en/stable/reference/userapi.html)

¹³ [io.imshow: https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imshow](https://scikit-image.org/docs/dev/api/skimage.io.html#skimage.io.imshow)

¹⁴ [pyplot.imshow: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html)

¹⁵ [np.array: https://numpy.org/doc/stable/reference/generated/numpy.array.html](https://numpy.org/doc/stable/reference/generated/numpy.array.html)

¹⁶ [Guía de Usuario: https://scikit-image.org/docs/dev/user_guide.html](https://scikit-image.org/docs/dev/user_guide.html)

¹⁷ [skimage.data: https://scikit-image.org/docs/stable/api/skimage.data.html](https://scikit-image.org/docs/stable/api/skimage.data.html)



Figura 1: lena.png.

Ejercicio 1.1.- Leer y visualizar imágenes

- a) Lea la imagen 'lena.png' (Figura 1); llamémosle `lena1`. Escriba por pantalla su tamaño (`shape`), sus valores mínimo y máximo, y su tipo. Visualice la imagen.
- b) Lea la imagen 'lena.png', usando el parámetro '`as_gray=True`'; llamémosle `lena2`. Escriba por pantalla su tamaño, sus valores mínimo y máximo, y su tipo. Visualice la imagen, consiguiendo que se vea en escala de grises.
- c) Divida el valor de la imagen `lena2` por 2 y súmele 0,25; llamémosle `lena3` al resultado. Escriba por pantalla sus valores mínimo y máximo. Visualice en subfiguras anexas `lena2` y `lena3`, usando escala de grises, y consiguiendo que las diferencias de brillo sean apreciables visualmente.

Ejercicio 1.2.- Escribir imágenes en disco

- a) Guarde las imágenes `lena2` y `lena3` del Ejercicio 1.1, en los ficheros `lena2.png` y `lena3.png`, respectivamente, de la carpeta `resultados`. Asegúrese de que la diferencia de intensidades es apreciable en las imágenes guardadas (con un visor externo, o volviéndolas a leer).
- b) Repita la operación anterior, pero multiplicando antes las imágenes por 10, y almacenando los resultados en `lena2b.png` y `lena3b.png`, respectivamente. ¿Qué ocurre y por qué? ¿Cómo debemos hacer para que las diferencias en las imágenes se puedan almacenar?

Ejercicio 1.3.- Enteros, Flotantes y Booleanos

Usando la siguiente función, que crea imágenes de degradado en escala de grises, con un tamaño, tipo y rango dados:

```
import numpy as np

def intensity_gradient(size=(100, 100), dtype=np.uint8,
                      values=(0, 255)):
    rows, cols = size
    mn, mx = values
    result = np.linspace(mn, mx, rows*cols).reshape(rows, cols)

    return result.astype(dtype)
```

- a) Cree tres imágenes de degradado, de tamaño 100×100 , y:
 - A: tipo `np.float64`, y valores de 0.1 a 0.9.
 - B: tipo `np.uint8`, y valores de 25 a 230.
 - C: tipo `np.int32`, y valores de -230 a 230.
- b) Visualice las tres imágenes, en escala de grises y sin normalizar, y reporte su valor mínimo y máximo.
- c) Explore el resultado de las funciones `img_as_float`, `img_as_uint8`, `img_as_int` e `img_as_bool` de `skimage` sobre las tres imágenes anteriores, así como la salida de la función `skimage.dtype_limits` aplicada sobre las imágenes de entrada y salida. Explique lo que ocurre.

Ejercicio 1.4.- Indexación

Considerando las imágenes `data.brick()` y `data.astronaut()`, convertidas a números flotantes en un rango apropiado:

- a) Sobre una copia de `brick`, asigne un valor de intensidad "blanco puro", a la ventana definida por las filas 100 a 150, y las columnas 20 a 120. Visualice el resultado, su valor máximo, su valor mínimo y el tipo.
- b) Cree una máscara binaria (llamémosle `brickmsk`) que valga 1 en las posiciones donde `brick` sea mayor que el 50 % del rango dinámico y 0 en el resto. Visualice el resultado, así como su valor máximo, su valor mínimo y el tipo.
- c) Modifique los valores de una copia de `astronaut` para tomen un valor `[r,g,b]` aleatorio en las posiciones indicadas por la máscara `brickmsk`^a
- d) Usando una copia de `astronaut`, visualice, en cuatro subfiguras, cada uno de sus canales RGB, en escala de grises, junto con la imagen en color.

- e) Usando la función `skimage.draw.disk`^b, obtenga las coordenadas de un círculo en una posición aleatoria del espacio 512×512 , y radio 20. Modifique la imagen resultante del apartado (c) para que los valores del canal verde en las posiciones del círculo sean 100 % del rango dinámico, dejando el resto de valores sin cambiar. Visualice el resultado.

^aNote que puede usar una máscara para indexar una imagen, si coinciden los tamaños de las dimensiones indexadas

^b`skimage.draw.disk`: <https://scikit-image.org/docs/stable/api/skimage.draw.html#skimage.draw.disk>

2. Transformación del espacio de color

El módulo `skimage.color`¹⁸ proporciona funciones para la transformación del espacio de color, como pueden ser: `rgb2gray`, `rgb2lab`, `rgb2hsv`, entre otros.

Ejercicio 2.1.- Espacios de color

Supongamos que queremos segmentar la vasculatura de la imagen `data.retina()`, y segmentar la mariposa de la imagen `butterfly.jpg`. Considerando cada una de estas imágenes:

- Muestre en una cuadrícula de 3×3 de subfiguras: los tres canales de RGB en escala de grises, los tres canales de CIE-L*a*b* en escala de grises, y los tres canales de HSV en escala de grises. Use los títulos de las figuras para indicar el nombre del canal y su rango de valores.
- Observando los resultados: ¿Qué canal es más apropiado para la segmentación objetivo? Razone la respuesta.

Ejercicio 2.2.- Canal de color arbitrario

- Haga una función `rand_channel` que, tomando como entrada una imagen de tres canales, obtenga como salida una imagen de un solo canal que resulta de la combinación lineal aleatoria de los tres canales.

Es decir, si la imagen de entrada I es una matriz $N \times M \times 3$, y W es un vector de 3 pesos aleatorios tal que $\sum W_i = 1$ ^a, lo que buscamos es la matriz de salida O , de $N \times M$ elementos, que resulta de hacer producto escalar entre I y W a lo largo de la tercera dimensión de I (e.g. con `np.dot`^b o `np.matmul`^c).

- Utilice la función `rand_channel` para visualizar 6 canales aleatorios, usando como entrada las imágenes del Ejercicio 2.1.

¿Es alguno de estos canales aleatorios "mejor" que cualquiera de los espacios de color del Ejercicio 2.1 para alguno de los casos? Razone la respuesta.

^aBasta tomar 3 números aleatorios y dividirlos por su suma

^b`np.dot`: <https://numpy.org/doc/stable/reference/generated/numpy.dot.html>

^c`np.matmul`: <https://numpy.org/doc/stable/reference/generated/numpy.matmul.html>

¹⁸`skimage.color`: <https://scikit-image.org/docs/dev/api/skimage.color.html>

3. Histogramas y ajustes de luminosidad y contraste

En **Scikit-image** disponemos del módulo `skimage.exposure`¹⁹, que proporciona funciones para el ajuste de intensidad y contraste de imagen.

La representación del histograma de una imagen es de especial relevancia para analizar la distribución de valores de la misma. Existen diversas utilidades en las librerías utilizadas que nos permitirán obtener dicha representación, como pueden ser: `exposure.histogram`²⁰, `np.histogram`²¹, `ndi.histogram`²², etc. o `plt.hist`²³ para visualización.

Ejercicio 3.1.- Ajuste de contraste global

Considerando las imágenes `data.text()` y `data.page()`, represéntelas en escala de gris junto con su histograma. Compare los resultados de las siguientes técnicas de ajuste de contraste global. Visualice los resultados, junto con sus histogramas.

- a) **Modificación lineal del rango dinámico.** Usando la función `rescale_intensity`, y expandiendo el histograma para que ocupe todo el rango dinámico disponible.
- b) **Window level contrast enhancement.** Usando la función `rescale_intensity`, y seleccionando un rango de entrada a mano que permita maximizar el contraste entre texto y fondo.
- c) **Ajuste Gamma.** Usando la función `adjust_gamma`, y seleccionando un valor de `gamma` apropiado que maximice el contraste entre texto y fondo.
- d) **Ajuste logarítmico.** Usando la función `adjust_log`, y seleccionando unos parámetros que maximicen el contraste entre texto y fondo.
- e) **Ecualización de histograma.** Usando la función `equalize_hist`.

Discuta los beneficios o inconvenientes de cada una de las técnicas.

Ejercicio 3.2.- Ajuste de contraste adaptativo

Considerando las imágenes `data.text()` y `data.page()`, represéntelas en escala de gris junto con su histograma. Compare los resultados de las siguientes técnicas de ajuste de contraste local. Visualice los resultados, junto con su histograma.

- a) **Ecualización adaptativa de histograma.** Usando la función `equalize_adapthist`. Explore el efecto del parámetro `kernel_size`. ¿Cómo influye en el resultado?
- b) **Control adaptativo de contraste.** Implemente una función que realice el control adaptativo de contraste lineal visto en clase. Puede usar la función `local_meanstd`, definida a continuación, para calcular las medias y desviaciones típicas locales. ¿Cómo influye el parámetro `sigma` en el resultado?

¹⁹`skimage.exposure`: <https://scikit-image.org/docs/stable/api/skimage.exposure.html>

²⁰`exposure.histogram`: <https://scikit-image.org/docs/stable/api/skimage.exposure.html#skimage.exposure.histogram>

²¹`np.histogram`: <https://numpy.org/doc/stable/reference/generated/numpy.histogram.html>

²²`ndi.histogram`: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.histogram.html>

²³`plt.hist`: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

Discuta los beneficios o inconvenientes de estas técnicas con respecto a los métodos globales.

```
import scipy.ndimage as ndi
import numpy as np

def local_mean_std(image, sigma=20):
    # Compute the local mean and std using Gaussians
    image = image.astype(np.float64)
    local_mean = ndi.gaussian_filter(image, sigma, mode='mirror')
    local_std = ndi.gaussian_filter(image**2, sigma, mode='mirror')
    local_std -= local_mean**2
    local_std = np.sqrt(local_std) + np.finfo(np.float64).eps
    return local_mean, local_std
```

Ejercicio 3.3.- Ajuste de contraste en color

Considere las imágenes de ejemplo `data.retina()` y `data.colorwheel()`. Apliqueles ecualización de histograma, mostrando los histogramas de entrada y salida para cada canal RGB por separado. ¿Por qué cambia la tonalidad? ¿Cómo podríamos hacer para ecualizar la luminosidad sin cambiar la tonalidad? Visualice ejemplos.

4. Filtrado espacial y eliminación de ruido

En el módulo `scipy.ndimage` disponemos de conjunto de filtros espaciales ya implementados. Los filtros lineales se fundamentan en el uso directo del operador de convolución en 1D en cada dimensión (`ndi.convolve1d`²⁴), si el filtro es separable, o en el operador de convolución 2D (`ndi.convolve`²⁵), para el caso general. Los filtros no lineales y de orden, en cambio, requieren implementaciones específicas de procesamiento por ventanas.

Exploraremos algunos de estos filtros para eliminación de ruido ya presente en la imagen, o incorporado artificialmente. En **Scikit-image**, disponemos de la utilidad `random_noise`²⁶, que nos permitirá generar diferentes tipos de ruido.

Ejercicio 4.1.- Filtro de medias

- a) Implemente una función que permita hacer un filtrado de medias sobre una imagen de entrada usando operaciones de convolución^a. La función debe tener la siguiente cabecera y parámetros:

²⁴`ndi.convolve1d`: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve1d.html>

²⁵`ndi.convolve`: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve.htm>

²⁶`random_noise`: https://scikit-image.org/docs/dev/api/skimage.util.html#skimage.util.random_noise

```
def box_filter(image, size=3, mode='nearest'):
    """Perform a mean filter using a square footprint of given
    ↪ size

    Keyword arguments:
    image -- the input image
    size -- the size of the box filter
    mode -- the convolution mode for boundary management
    """
    ...
    return ...
```

- b) Visualice el resultado de aplicar la función `box_filter` sobre la imagen `data.coins()`, usando diferentes tamaños de ventana.

^aEl filtro de medias es separable

Ejercicio 4.2.- Eliminación de ruido: estudio comparativo

Para la imagen `data.camera()` (en escala de grises):

- Altere la imagen con **ruido blanco^a Gaussiano aditivo**. Seleccione dos valores para el parámetro de *varianza* (uno alto y otro bajo), de forma que el ruido sea apreciablemente diferente entre ambos, y con respecto a la imagen original, pero sin perder toda la información de la imagen. Visualice los resultados.
- Altere la imagen con **ruido "sal y pimienta"**. Seleccione dos valores para el parámetro de *cantidad* (uno bajo y otro alto), de forma que el ruido sea apreciablemente diferente entre ambos, y con respecto a la imagen original, pero sin perder toda la información de la imagen. Visualice los resultados.
- Escriba un código que aplique los filtros `ndi.gaussian_filterb` y `ndi.median_filterc` a la imagen original, y a las resultantes de los apartados (a) y (b), mostrando los resultados de forma ordenada.
- Explore el efecto de los parámetros de tamaño (`size` o `sigma`, según el caso) en los resultados del apartado (c), mostrando ejemplos representativos.

Discuta las ventajas e inconvenientes de los filtros comparados a la vista de los resultados, teniendo en cuenta cómo afectan sus parámetros.

^aDe media 0

^b`ndi.gaussian_filter`: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.htm

^c`ndi.median_filter`: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.median_filter.html

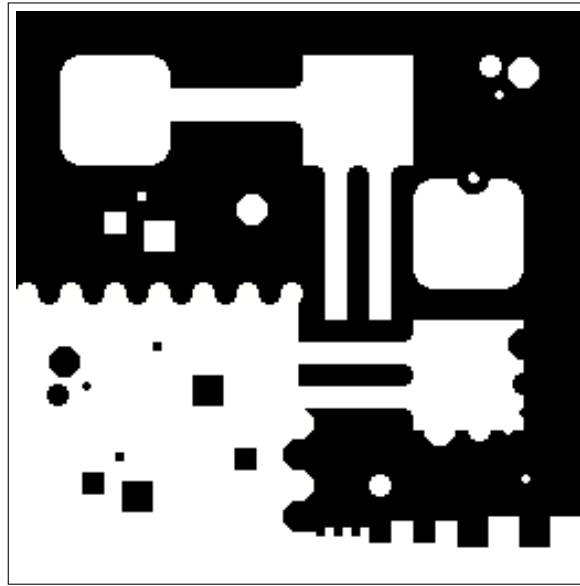


Figura 2: Imagen morph256.png

5. Operadores morfológicos

El módulo `skimage.morphology`²⁷ proporciona un conjunto de operadores morfológicos y utilidades asociadas.

Ejercicio 5.1.- Morfología binaria

Aplique las siguientes operaciones morfológicas a la imagen `morph256.png` (Figura 2), utilizando elementos estructurantes cuadrados (`morphology.square`) y circulares (`morphology.disk`) de tamaños 5, 10 y 15:

- **Erosión:** `morphology.erosion`
- **Dilatación:** `morphology.dilation`
- **Apertura:** `morphology.opening`
- **Cierre:** `morphology.closing`

- a) Visualice los resultados de forma ordenada.
- b) ¿Qué diferencias se aprecian?

Ejercicio 5.2.- Morfología en escala de grises

Repita los resultados del Ejercicio 5.1 con la imagen `data.camera()`. Visualice los resultados de forma ordenada y explique qué efectos tiene cada uno de los operadores.

²⁷`skimage.morphology`: <https://scikit-image.org/docs/stable/api/skimage.morphology.html>

6. Operadores de derivada y detección de bordes

La función `ndi.gaussian_filter` de `scipy.ndimage` permite calcular filtros de *Derivada de Gaussiana* usando el parámetro `order`. Esto es equivalente a aplicar un suavizado Gaussiano, seguido de un operador de derivada. Para derivadas de primer orden, el efecto es similar a un filtro de Sobel con tamaño controlado por `sigma`²⁸.

Para calcular la derivada de orden `n` en la dirección `x`, de las columnas, basta con especificar `order=[0,n]`. En el caso de la dirección `y`, de las filas, usaremos `order=[n,0]`.

Ejercicio 6.1.- Visualización de filtros efectivos

- Cree una **imagen delta** de 101×101 pixels y tipo flotante, que tome el valor 0 en todas las posiciones, excepto en la posición central, que tomará el valor 1. La convolución de cualquier filtro con esta imagen resultará en el mismo filtro, lo que nos permitirá ver su forma.
- Use la función `ndi.gaussian_filter` sobre la imagen **delta**, calculando la **primera derivada de Gaussiana** en vertical y horizontal, para valores de `sigma` de 5, 10 y 20 pixels. Visualice los resultados, teniendo en cuenta que tendrán valores positivos y negativos.
- Use la función `ndi.gaussian_filter` sobre la imagen **delta**, calculando la **segunda derivada de Gaussiana** en vertical y horizontal, para valores de `sigma` de 5, 10 y 20 pixels. Visualice los resultados, teniendo en cuenta que tendrán valores positivos y negativos.

Ejercicio 6.2.- Gradiente, magnitud y orientaciones

Para las imágenes `circles.png`, `circlegrad.png` y `data.camera()`:

- Estime el gradiente de la imagen calculando sus **derivadas en x e y** para un `sigma` determinado. Visualice las componentes del gradiente, teniendo en cuenta que tendrán valores positivos y negativos.
- Calcule la **magnitud del gradiente**, y visualice el resultado.
- Calcule la **orientación del gradiente**, usando `np.arctan2`, y visualice el resultado, teniendo en cuenta que estarán en el rango $[-\pi, \pi]$.

¿Cómo afecta el parámetro `sigma` a los resultados?

Ejercicio 6.3.- Laplaciano de Gaussiana

Implemente una función que **calcule el Laplaciano usando las segundas derivadas de Gaussiana**. Aplíquelo a las imágenes `circles.png`, `data.camera()` y `delta` (calculada como en el Ejercicio 6.1). Visualice los resultados, teniendo en cuenta que tendrán valores positivos y negativos. ¿Qué caracteriza a los bordes de la imagen en esta representación? ¿Cómo afecta el parámetro `sigma` a los resultados?

²⁸Podríamos obtener un filtro con efecto similar a los de Prewitt, y escala arbitraria, combinando un filtro de medias de un tamaño dado con el operador de derivada.

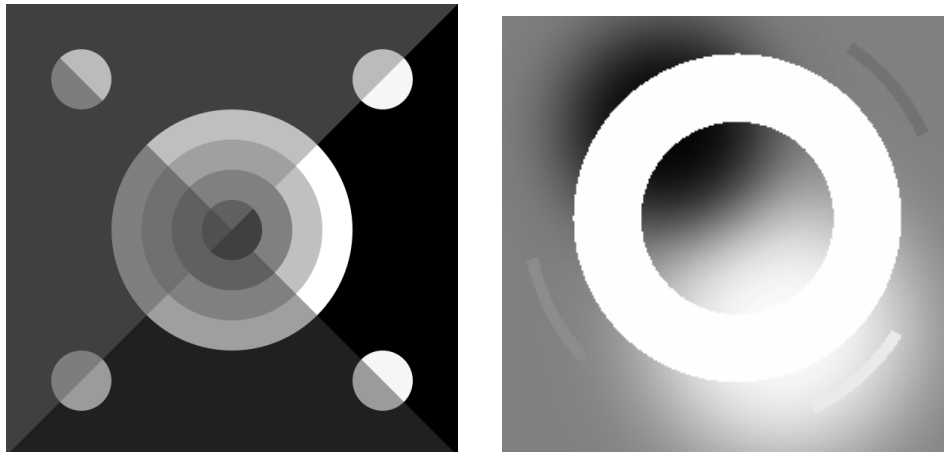


Figura 3: Imágenes `circles.png` (izquierda) y `circlegrad.png` (derecha)

Algoritmo de Canny

El submódulo `skimage.feature`²⁹ dispone de una implementación del algoritmo de Canny (`feature.canny`³⁰).

Ejercicio 6.4.- Algoritmo de Canny

Aplique el **algoritmo de Canny** a las imágenes `circlegrad.png` y `data.coins()`, ajustando su parámetro `sigma` y los umbrales de histéresis, de forma que se obtengan los mejores bordes posibles (los bordes del aro y los contornos de las monedas). Visualice los resultados.

Reomendaciones para el ajuste:

- **Umbral bajo.** Manteniendo los dos umbrales iguales, busque un valor lo suficientemente bajo que aparezcan, al menos, todos los bordes requeridos para un `sigma` adecuado (alta sensibilidad, baja especificidad).
- **Umbral alto.** Manteniendo los dos umbrales iguales, busque un valor que solo detecte bordes requeridos, aunque no sean todos (alta especificidad, baja sensibilidad).
- **Ajuste fino.** Usando la combinación de umbrales anteriores, realice un ajuste fino, o vuelva a iterar sobre el proceso probando un `sigma` diferente.

7. Segmentación

Umbralización de imágenes

En el módulo `skimage.filters` disponemos de diferentes utilidades para umbralización automática de imágenes.

²⁹`skimage.feature`: <https://scikit-image.org/docs/stable/api/skimage.feature.html>

³⁰`feature.canny`: <https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.canny>

Ejercicio 7.1.- Umbralización global

En las imágenes `data.text()` y `data.page()` queremos separar el texto del fondo.

- a) Visualice el histograma de las imágenes.
- b) Obtenga un umbral global mediante los siguientes métodos, y represente estos umbrales sobre el histograma (e.g. con líneas verticales de varios colores).
 - Método de Otsu. `threshold_otsu`^a
 - Método del triángulo. `threshold_triangle`^b
 - Método de ISODATA. `threshold_isodata`^c
 - Método manual. Usando un umbral global ajustado a mano por usted.
- c) Visualice la segmentación resultante con cada uno de estos umbrales globales.

Discuta la conveniencia o no de cada uno de estos métodos para la segmentación de las imágenes.

^a`threshold_otsu`: https://scikit-image.org/docs/stable/api/skimimage.filters.html#skimimage.filters.threshold_otsu
^b`threshold_triangle`: https://scikit-image.org/docs/stable/api/skimimage.filters.html#skimimage.filters.threshold_triangle
^c`threshold_isodata`: https://scikit-image.org/docs/stable/api/skimimage.filters.html#skimimage.filters.threshold_isodata

Ejercicio 7.2.- Umbralización adaptativa (local)

Calcule mapas de umbrales locales usando la función `threshold_local`^a para segmentar las imágenes `data.page()`, donde queremos separar el texto del fondo, y `butterfly.png` (en escala de gris), donde queremos segmentar la mariposa. Visualice los resultados.

Compare los métodos `'mean'`, `'median'`, y `threshold_otsu` (usando `'generic'` y pasando la función como parámetro). ¿Cómo influye el parámetro de tamaño de bloque en el resultado?

^a`threshold_local`: https://scikit-image.org/docs/stable/api/skimimage.filters.html#skimimage.filters.threshold_local

Componentes conexas

El módulo `skimage.measure`³¹ proporciona un conjunto de funciones útiles para identificar objetos segmentados en la imagen y obtener medidas significativas de los mismos.

Ejercicio 7.3.- Análisis de componentes conexas

La función `data.binary_blobs`^a permite generar imágenes aleatorias similares a la de la Figura 4. Las usaremos como imágenes de entrada, variando los parámetros `blob_size_fraction` y `volume_fraction`.

³¹`skimage.measure`: <https://scikit-image.org/docs/stable/api/skimimage.measure.html>

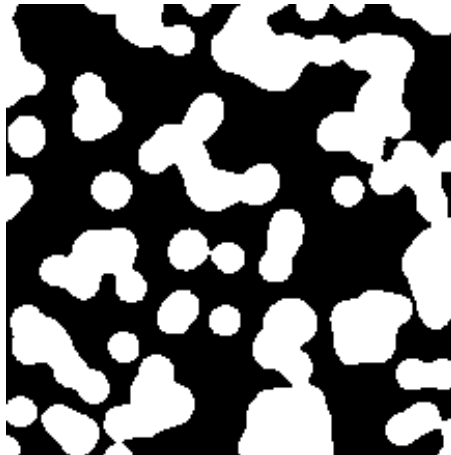


Figura 4: Imagen de blobs aleatorios

Sobre estas imágenes:

- a) Use la función `measure.label`^b, para etiquetar cada una de las componentes conexas en la imagen, y visualicelas de forma que cada objeto aparezca de un color o nivel de gris apreciablemente diferente.
- b) Con el resultado de la función anterior, y haciendo uso de la función `measure.regionprops`^c, imprima por pantalla el número de componentes conexas, así como la media y desviación típica de sus áreas.

^a`data.binary_blobs`: https://scikit-image.org/docs/stable/api/skimage.data.html#skimage.data.binary_blobs

^b`measure.label`: <https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.label>

^c`measure.regionprops`: <https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops>